Open in app ↗

Search

Time in Distributed Systems

Suresh Podeti
3 min read · May 2, 2023

Listen Share More

Physical clocks which we use in real life are based on **measurement of some physical phenomenon**. Examples of physical clocks include: Pendulum clock, Hour glass clock, Quartz clock, Atomic clocks.

All these devices rely on physical processes to measure time. Of course, **there can be errors** residing both in **measurement tools** being used and the actual **physical processes** themselves.

The main use cases of time in a software system is to determine the **order between different events.**

In a **centralised system**, we have **only a single node** thus we have o**nly a single clock**. This means one can maintain the illusion of single, universal time dimension which can determine the order of the various events in the single node of the system.

There is **no global time** in distributed system, because **each node has** its **own clock** and each one of those clocks may run at a different rate, which means they **will drift apart** from each other. No matter how often we synchronise these clocks with each other or with other clocks using **NTP** (Network time protocol), there **will always be a skew** between various clocks involved in a distributed system.

Let's assume we have a distributed system composed of 3 different nodes A, B, and C. Every time an event happens at a node, the node assigns a time stamp to the event, using its own clock, and then propagates this event to the other node. As, the nodes receive events from the other nodes, they compare the timestamps associated with these events to determine the order in which the events happened. Assume, there is a **skew between the clocks of the various nodes**, the **correctness of the system is violated.**

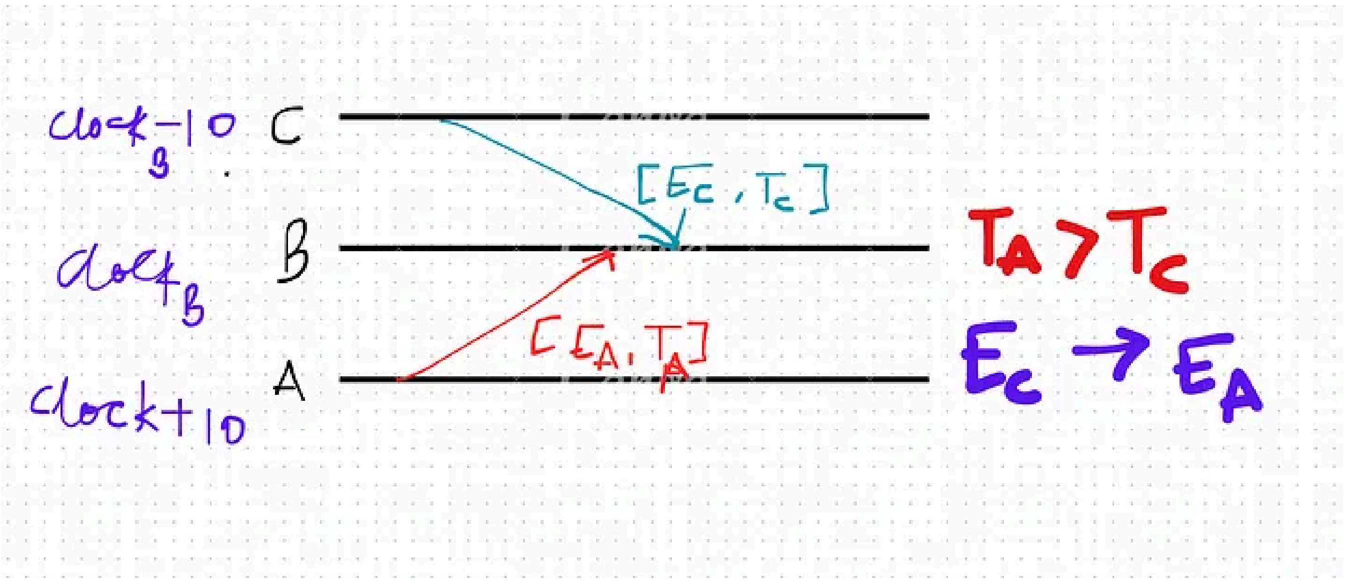

Figure: Side-effects of assuming a global clock

We assume that the clock of node A is running ahead of the clock of node B. In the same way, the clock of node C is running behind the clock of the node B. As a result, **even if the event in node A happened before the event in node C, node B will compare the associated timestamps and will believe event from node C happened first.**

Logical Clocks

These clocks **do not rely on physical processes** to keep track of time.

Instead, they make use of messages exchanged between the nodes of the system, which is main mechanism information flows in a distributed system. Logical clocks make use of **counters**, which would be **incremented** subsequently.

For example, if the system started at 9:00 and events A, B, C happened at 9:01, 9:05, and 9:55 respectively, then they could be assigned the timestamps 1, 2, 3. As a result, the **system would still be able to order the events**, but it **would not be able to determine the temporal distance between any two events**.

That's all of this article, I will explain you how this logical clocks are used to order events in distributed systems.

Distributed Systems | Event Order | Logical Clock | Lamport Clock | Version Vector

Edit profile

Written by Suresh Podeti

1.2K Followers

Aspiring Software Architect | SDE III (Golang) @JungleeGames| Ex-Byju's | Co-founder @Rupant Tech. | Ex-Synopsys | I.I.T Kharagpur | 7+ Years of Experience

Recommended from Medium